

## ESQUEMA AULA PRÁTICA 1

### ❑ Familiarização com o Ambiente JBuilder

### ❑ Introdução à Linguagem de Programação JAVA

**0** – Inicie o ambiente de desenvolvimento integrado JBuilder.

Start|Programs...|JBuilder 6 Professional

Copie do notepad toda a chave do software. Feche o notepad.

No assistente de instalação escolha Have license key e faça <CTRL+V>.

**1** – Usando o editor do ambiente de desenvolvimento JBUILDER pretende-se construir e executar o programa abaixo.

```
class Primeiro {  
    public static void main(String[] args) {  
        System.out.println("Este é o 1º programa em Java");  
    }  
}
```

Terá que começar por criar um projecto `file|new project...`

O ficheiro de projecto (com extensão `jpx` ou `jpr`) mantém definições relevantes para o novo projecto e uma lista de todos os ficheiros que o constituem. Pode incluir a estrutura de directorias, caminhos para as bibliotecas, código fonte e versões. O Jbuilder acede a esta informação sempre que carrega, guarda, modifica ou recompila um projecto.

No passo 1 do assistente (wizard) indique o nome e a directoria base do seu projecto.

**Deverá indicar uma localização na sua conta na máquina *ciunix* ou *alpha4* (se desejar recuperar posteriormente o trabalho realizado!).**

Preencha os campos do passo 3 do assistente.

- Um programa em JAVA é uma classe.
- As classes são agrupadas em “pacotes” (packages).
- Cada aplicação terá que possuir uma classe onde está definido o método **main** pelo qual se inicia a execução do programa.
- Por convenção o nome de uma classe começa por uma letra maiúscula.
- **;** é um terminador de instruções.
- **{ }** delimitador de conjuntos de instruções.

Deve pois criar uma classe dentro deste projecto (`file|new class`).

No assistente “baptize” a nova classe com o nome `Primeiro`. Pode desmarcar as opções apresentadas pelo assistente.

Escreva o código do programa.

Para executar o programa comece por compilá-lo (`Project|Make project`), e seguidamente executá-lo (`Run|project`) — alternativamente pode utilizar a combinação de teclas `<CTRL+F9>` seguida de `<F9>`.

**Atenção:** Antes da primeira execução do programa terá que indicar a classe que contém o método `main`.

**Main classe:** prima o botão reticências e escolha `nomedopacote.Primeiro`

### **Notas:**

O ficheiro que contém a classe JAVA é gravado com a extensão **.java**; A compilação deste ficheiro produz um outro de extensão **.class** que poderá ser interpretado pelo interpretador de JAVA da sua máquina.

**Uma ideia base da linguagem JAVA é a de que “um programa em JAVA deve poder ser executado em qualquer parte”. Assim o código fonte da aplicação é compilado para uma representação intermédia, independente do sistema de execução e da arquitectura da máquina. Essa representação intermédia é designada por byte-code.**

**De seguida este código pode ser interpretado sobre o ambiente de cada máquina específica. Para cada plataforma em que se pretende executar um programa em Java é necessário um “motor de execução” designado por Java Virtual Machine” (JVM). A JVM recebe byte-code e transforma-o em instruções executáveis na máquina onde o ambiente Java é instalado.**

**O JAVA pode ser usado para criar dois tipos de programas: Aplicações e Applets.**

**Applets são porções de código Java não executável por si próprio. Requerem a existência de um “browser” que incorpore e execute a JVM.**

**2 – Tipos de Dados Primitivos**

<b>Tipo</b>	<b>Valores</b>	<b>Valor por omissão</b>	<b>Nº de bits</b>	<b>Gama de valores</b>
<b>boolean</b>	true, false	false	1	--
<b>char</b>	caracteres unicode	\u0000	16	\u0000 a \uFFFF
<b>byte</b>	inteiro com sinal	0	8	-128 a +127
<b>short</b>	inteiro com sinal	0	16	-32768 a + 32767
<b>int</b>	inteiro com sinal	0	32	-2147483648 a +2147483647
<b>long</b>	inteiro com sinal	0	64	-1E+20 a +1E+20
<b>float</b>	IEEE 754 FP	0.0	32	±3.4E+38 a ±1.4E-45
<b>double</b>	IEEE 754 FP	0.0	64	±1.8E+308 a ±5E-324

**3 – Valores, variáveis e constantes****Declaração de variáveis:**

**<tipo da variável> <identificador da variável> [ =valor] [, ...] ;**

**Exemplos:**

int x;

int y =10;

char um ='1';

char newline = '\n'

boolean verdade;

verdade = true;

float f = 9.1234567

double d = 9.123456789012345

**Declaração de constantes:**

```
final float pi=3.14159273269;
```

**4 - Construa o seguinte programa:**

```
class Valores {  
    public static void main(String[] args){  
        int numero;  
        double decNum;  
        numero = -100000;  
        decNum = 12345.6789;  
        System.out.println("O valor da variável inteira é: " + numero);  
        System.out.println("O valor da var. real é: " + decNum);  
        char letra = 'Q';  
        System.out.println( letra);  
    }  
}
```

**5 – Conversão entre tipos**

É possível converter um dado tipo num outro compatível (!) usando um operador unário de “casting”.

Por exemplo:

```
...  
int x = 2;  
float f;  
f = (float) x; /* converte o valor inteiro de x no real 2.0 */  
...  
char c='A';  
int i;  
i = (int) c;
```

## **6 – Ler um carácter do teclado**

O método `System.in.read()` permite ler dados a partir de um buffer associado ao teclado.

## **7 – Construa o seguinte programa**

```
class LerCaracter {  
    public static void main(String[] args) throws java.io.IOException /* !!! */  
    {  
        char c;  
        System.out.print("Introduza um carácter pelo teclado: ");  
        c = (char)System.in.read();  
        System.out.println("O carácter que escreveu foi: " + c);  
    }  
}
```

## **8 – Estude os seguintes operadores**

- **aritméticos:** +, -, \*, /, %
- **incremento e decremento:** ++, --  
    **(qual a diferença entre `i2=i1++` e `i2=++i1` ? )**
- **operadores de atribuição:** +=, -=, \*=, /=, %=

## Para explorar ...

9 – Implemente agora o seguinte programa<sup>1</sup>:

```
import java.util.*;

public class Segundo {
    private static String[] saudacoes =
        {"Bom dia", "Boa tarde", "Boa noite"};
    private static String nome = "Escreva aqui o seu nome";
    private static int horas;
    private static int minutos;

    public static void getHoras( ){
        GregorianCalendar calend = new GregorianCalendar();
        horas = calend.get(Calendar.HOUR_OF_DAY);
        minutos = calend.get(Calendar.MINUTE);
    }
    private static int periodoDia(int h){
        return (h+20) /8 % 3;
    }
    public static void main(String args[]){
        getHoras ( ) ;
        System.out.println(saudacoes[periodoDia(horas)] + ", " + nome);
        System.out.println("Passam " + minutos + " minutos das " +
            horas + " horas." ) ;
    }
}
```

Depois de analisar o programa, e apesar de ainda não ter conhecimentos para o perceber na totalidade, tente modificá-lo para que:

- i) Apresente as horas de um modo gramaticalmente correcto (1 hora, 1 minuto!  
Passam, faltam! Horas exactas?)
- ii) Apresente também a data.
- iii) Indique o número de dias que faltam até ao fim-de-semana...

Apresente a listagem das datas das 13 próximas Sextas-feiras 13.

<sup>1</sup> Adaptado de “A small cup of Java” de Pedro Guerreiro